云聚合部署文档

总体描述

“云聚合”是白山云科技有限公司（以下简称：白山）推出的针对数据构建、适配，接口加速，接口防护在内的新一代云服务产品，可以有效的解决企业面临的数据开放、接口统一、接口速度卡顿、缓慢，CC攻击等问题，帮助企业更好的为客户服务。云聚合产品支持公有云、私有云两种模式部署，公有云模式直接通过在线平台开通账号即可使用，私有云模式支持将云聚合产品部署到企业内部去，进行内部使用。

CC防火墙是云聚合里进行接口保护的一个子产品，CC防火墙的特点包括：

- 软件实现
- 旁路拦截
- 机器学习算法
- 大数据实时分析
- 内核级高速拦截
- 防御多种HTTP攻击类型

CC防火墙根据访问日志，通过Storm大数据流式分析平台，实时分析出攻击IP，CC防火墙包括拦截器，企业可以将拦截器部署在需要被保护的服务器上，从而保护这些服务器在遇到CC攻击时仍能保持服务器状态正常，并且有效提高业务服务。

硬件环境

Intel 24核+，48G+内存，500G+硬盘（无需做raid）

软件环境

操作系统

Linux CentOS 6.2以上

软件

Nginx 1.10.1
PHP-FPM 5.4.45-12
MySQL 5.6.33-2
PHP 5.4.45
Redis 3.2.3
HDP Storm 0.10.0
HDP Kafka 0.9.0
HDP ZooKeeper 3.4.6
elastic search 2.3.5-1

网络环境

1000Mb 网卡
内网互联，ping延迟低于5ms

角色

云聚合服务包括3各角色的节点，Web中心节点主要提供控制台UI，包括PHP+MySQL+Redis，在控制台UI上用户可以操作、查看云聚合的相关配置；分布式计算节点，主要由Storm集群组成，里

面包含了云聚合CC防火墙的核心算法，它对传过来的日志进行实时分析，并最终给出分析结果；拦截器，通过控制Kernel iptables进行精准HTTP层拦截，性能比WebService层拦截快一个数量级，拦截器需要部署在需要被保护的服务器上（通常是外层Proxy服务器，如nginx、ha proxy等）。

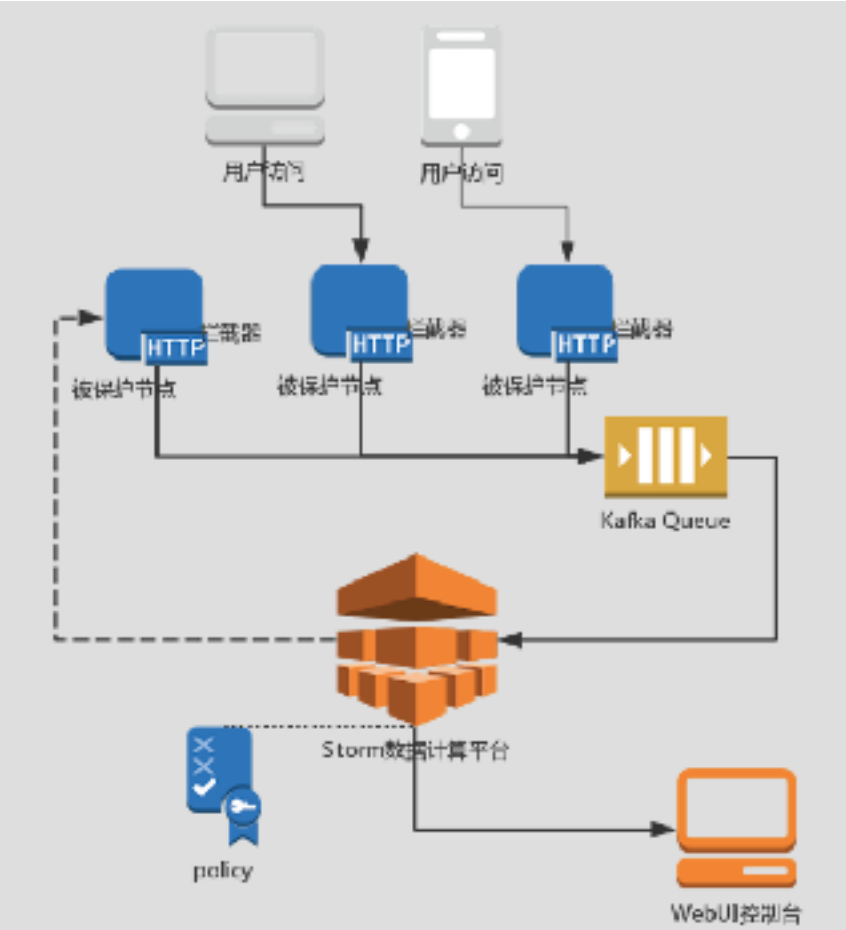
1，Web中心节点（最少1个节点）
- 用于提供REFTFul接口
- 用于提供云聚合操作界面

2，分布式计算节点（最少3个节点）
- 用于提供storm流式计算

3，ElasticSearch集群（最少3个节点）
- 用于存储日志等统计信息

4，拦截器（部署于需要被保护的服务器上）
- 用于进行实际IP拦截

架构图



云聚合CC防火墙架构图

如图所示，用户的请求落在HTTP节点上（即被保护的节点），这些HTTP请求的日志被推送到Kafka队列里，Kafka队列将数据输送到Storm大数据计算平台，然后计算平台根据算法分析出需要拦

截的IP，并且通知拦截器，拦截器最终对这些IP进行拦截。WebUI作为控制台，用户可以登录其上查看服务状态，并进行简单的配置。

服务对接

CC防火墙产品对接主要需要两部分：

1，将HTTP请求日志送入Kafka队列，可以通过kafkacat把对应格式的nginx访问日志推送到kafka集群（https://github.com/edenhill/kafkacat）

每一条HTTP日志是一个json格式的字符串，下面是一条日志信息：

{“remote_addr”:”123.125.71.42","remote_user":"-","time_local":"2016-09-21T10:47:17+08:00","scheme":"https","http_host":"mt3.gomemyc.com","api_id":"226","api_path_id":"1310","caller_id":"0","method":"GET","request_uri":"/loan/CBBB8389-6550-479D-A5D5-12AC08EDE4E8?go_back_path=offline-events","uri":"/loan/CBBB8389-6550-479D-A5D5-12AC08EDE4E8","request_time":"0.028","status":"200","upstream_addr":"118.26.23.180:80",","upstream_status":"200","upstream_response_time":"0.020","request_length":"292","body_bytes_sent":"9611","http_referer":"-","http_user_agent":"Mozilla/5.0 (compatible; Baiduspider/2.0; +http://www.baidu.com/search/spider.html)”,”http_x_forwarded_for":"-","upstream_cache_status":"MISS","hostname":"bgp-beijing-beijing-1-123-59-102-48"}

remote_addr： 客户端IP地址
remote_user：忽略
time_local： 请求时间
scheme：http或者https
http_host： http请求host
api_id： api id信息
api_path_id： 忽略
caller_id： 忽略
method： http method
request_uri： 请求的uri
uri： 请求的path
request_time： 请求时间
status： http请求返回给客户端的状态码
upstream_addr： 后端地址，可以忽略
upstream_status： 后端返回的http状态码，可以忽略
upstream_response_time： 请求后端的时间，可以忽略
request_length： 发送请求的长度
body_bytes_sent： 请求返回的长度
http_referer： http referer
http_user_agent： 客户端UA
http_x_forwarded_for： 可以忽略
upstream_cache_status： 忽略
hostname： 忽略

2，将拦截器部署在需要保护的节点上，拦截器进程为：tcp_force_reset

CC防火墙配置

CC防火墙配置采用XML文件格式，格式为：

```
<?xml version="1.0" encoding="UTF-8"?>
<hfw>
    <zkHosts>192.168.1.2:2181,192.168.1.3:2181</zkHosts>
    <topologyName>ccFirewall</topologyName>
    <topicName>juhe-log</topicName>
    <debug>off</debug> <!--onloff  -->
    <comment>
        <cip>corrent ip in 60s</cip>
        <cips>corrent ip in 60s</cip>
        <pv>http request</pv>
        <mpath>most path rate</mpath>
        <rlength>request size in bytes</rlength>
        <rtime>request time in second</rtime>
        <4xx>404 code number</4xx>
    </comment>
    <domain>
        <name>www.test.com</name>
        <enable>on</enable> <!--onloff  -->
        <expire>600</expire>
        <policy>
            <id>0</id>
            <name>HTTP-flood</name>
            <rule>cip[0].pv>10 and cip[0].mpath>0.9</rule>
            <action>ban</action>
        </policy>
        <policy>
        ……
        </policy>
    </domain>
    <domain>
    ……
    </domain>
    <actions>
        <action>
            <name>ban</name>
            <url>http://123.59.102.50:9000/ngproxy/rule/add</url>
        </action>
        <action>
            <name>none</name>
        </action>
        <action>
            <name>warn</name>
            <url>http://192.168.0.23:9000/ngproxy/rule/</url>
        </action>
    </actions>
</hfw>
```

下面是配置文件的具体说明：

<hfw> 是根标签，CC防火墙配置中只能有一个根标签。

<hfw> 包含以下标签：

1. 一个<zkHosts> 标签， CC防火墙从kafka中读数据，kafka的信息存放在zookeeper中，这个配置项就是kafka对应zookeeper地址信息，每一个地址信息格式为：ip:port，可以是一个或者多个地址信息，多个中间用逗号(,)隔开。
2. 一个<topologyName>标签，用来设置pology name。
3. 一个<topicName> 标签，用来设置kafka topic name。
4. 一个<debug>标签，设置是否开启debug模式，on表示开启，off表示关闭。
5. 一个<comment>标签，这个标签可有可无，用来做一些说明用。
6. 一个或者多个<domain>标签，每一个<domain>标签对应一个域名的防火墙配置。
7. 一个<actions>标签，CC防火墙执行的操作以及对应操作的接口地址

下面是<domain>标签的配置说明：

1. 一个<name>标签，配置CC防火墙保护的域名。
2. 一个<enable>标签，这个域名CC防火墙开关，on表示开启，off表示关闭。
3. 一个<expire>标签，配置封禁ip的过期时间。
4. 一个或者多个<policy>标签，配置策略信息，主要包括策略id(<id>) ，策略名称(<name>)，策略规则(<rule>)，以及符合这个规则触发的操作(<action>)，这个操作一定是<actions>中配置的某一项。

CC防火墙最核心就是<rule>标签的配置，rule规则判断可以确定某一个ip是否为攻击ip。
CC防火墙是针对某一个域名在一段时间内的ip行为特征进行分析，根据rule判断这个ip是否为攻击ip。CC防火墙rule就是对ip行为特征做判断。一个ip特征值表达式是这样：cip[begin:end].feature，其中cip代表当前IP的行为，begin为开始下标，end为结束下标，feature为特征值名称。cip[0]表示第0个值，存放的是最新一分钟的特征值集合。cip[1]表示第1个值，存放的是前一秒往前推一分钟的特征值集合。cip[10]表示第10个值，存放的是前10秒往前推一分钟的特征值集合。cip[0:10]，表示这段时间特征值的平均值。

cip对象包含的特征有：

1. pv：访问次数
2. 4xx：404错误的次数
3. mpath： 访问最多路径占比
4. rlength： 一段时间发请求的总长度，单位为字节
5. rtime： 一段时间请求的总时长，单位为秒

一条rule规则就是一个逻辑表达式(LExpression)，这个表达式返回值是true或者false，逻辑表达式由比较表达式(CExpression)和逻辑运算符构成，比较表达式是由算术表达式(AExpression)和比较运算符构成，算术表达式由常量，变量，算术运算符构成。下面详细说明：

LExpression = CExpression (and | or LExpression)?
CExpression = AExpression >|< AExpression
AExpression = VariableIConstant (+|-||/ VariableIConstant)?*
Constant = 数字常量
Variable = objectname[begin:end].feature

LExpression可以是一个CExpression，或者是 CExpression + 逻辑运算符 + 一个或者多个LExpression组成，这里的逻辑运算符支持 and 和 or 两种操作。

CExpression 组成为：AExpression + 比较运算符 + AExpression，支持的比较运算符有大于(>)和小于(<)符号。

AExpression可以是一个常量、变量，或者是VariableIConstant + 算术运算符 + VariableIConstant，支持的算术运算符有加(+)减(-)乘(*)除(/)

下面是一些例子：

1. cip[0].pv > 100 ，表示一个ip当前一分钟的pv大于100时返回true，否则返回false
2. cip[0].pv > 100 and cip[0].mpath > 0.9，表示一个ip当前一分钟pv大于100，并且当前一分钟的most path占比超过90%,返回true
3. cip[0].rtime/cip[0].pv > 1，当前一分钟平均每个请求的时长大于1秒返回true
4. cip[0].rlength/cip[0].pv > 1024，当前一分钟平均请求的长度大于1024字节返回true

运维手册

Web中心节点：

1，管理平台

启动：

```
# /etc/init.d/nginx start
# /etc/init.d/php-fpm start
```

停止：

```
# /etc/init.d/nginx stop
# /etc/init.d/php-fpm stop
```

重启：

```
# /etc/init.d/nginx restart
# /etc/init.d/php-fpm restart
```

检查服务状态：

```
# /etc/init.d/nginx status
# /etc/init.d/php-fpm status
```

主页地址：http://10.143.119.104/#/user/login

目前有效的用户名和密码：test@baishancloud.com 123456

2，iptables api

程序使用supervisord管理，具体启动命令见supervisord配置文件

启动：

```
# /etc/init.d/supervisord start
```

停止：

```
# /etc/init.d/supervisord stop
```

重启：

```
# /etc/init.d/supervisord restart
```

检查服务状态：

```
# /etc/init.d/supervisord status
# supervisorctl
cc_pushlog                       RUNNING   pid 20084, uptime 3:12:42
ngproxymod-iptablesapi           RUNNING   pid 19965, uptime 3:14:55
```

Storm Kafka ZooKeeper节点：

启动，停止，重启，检查服务状态等操作在Ambari管理平台操作，

地址：http://10.143.119.101:8080/，用户名：admin，密码：admin

拦截器：

需要部署到需要保护的服务器上（如nginx）：
启动：当有攻击时，由iptableapi接口来启动
停止：每天定时停止不起作用的程序
检查服务状态：进程存活

kafkacat：

需要部署到需要保护的服务器上（如nginx）：
程序使用supervisord管理，具体启动命令见supervisord配置文件
启动：
    # /etc/init.d/supervisord start
停止：
    # /etc/init.d/supervisord stop
重启：
    # /etc/init.d/supervisord restart
检查服务状态：
    # /etc/init.d/supervisord status

ES集群：

启动：
    /etc/init.d/elasticsearch start
停止：
    /etc/init.d/elasticsearch stop
重启：
    /etc/init.d/elasticsearch restart
检查服务状态：
    /etc/init.d/elasticsearch status

角色服务器对应关系（此对应关系随着真实环境而变化）：

服务器ID	服务器IP（举例参考）	角色
A	10.143.119.101	ZooKeeper+Storm+Kafka+ES
B	10.143.119.102	Zookeeper+Storm+Kafka+ES
C	10.143.119.103	ZooKeeper+Storm+Kafka+ES
C	10.143.119.104	Web中心节点

常见问题排查FAQ

1，如何判断防火墙工作状态?
通过storm的管理界面可以看到防火墙当前状态，如果有报错，查看storm日志排查问题

2，如何查看防火墙的日志延迟程度?

查看日志可以看到，防火墙每秒钟会打印一条当前正在执行的日志，日志中有时间信息，可以和当前时间做对比。

3，修改CC防火墙配置文件需要重启服务吗?
答：不需要，修改后，一分钟内生效

4，防火墙关闭会影响业务吗?
答：不会，防火墙的输入是通过日志异步推入，防火墙的输出最终作用在被保护的Linux服务器的iptabes上，这样，即使防火墙关闭，不会影响原有的业务程序和流程。

附：配置文件样例

```
<?xml version="1.0" encoding="UTF-8"?>
<hfw>
    <zkHosts>192.168.1.2:2181,192.168.1.3:2181</zkHosts>
    <topologyName>ccFirewall</topologyName>
    <topicName>juhe-log</topicName>
    <debug>off</debug> <!--onloff  -->
    <comment>
        <cip>corrent ip in 60s</cip>
        <pv>http request</pv>
        <mpath>most path rate</mpath>
        <rlength>request size in bytes</rlength>
        <rtime>request time in second</rtime>
        <n4xx>404 code number</n4xx>
    </comment>
    <domain>
        <name>www.test.com</name>
        <enable>on</enable> <!--onloff  -->
        <expire>600</expire>
        <policy>
            <id>0</id>
            <name>HTTP-flood</name>
            <rule>cip[0].pv>10 and cip[0].mpath>0.9</rule>
            <action>ban</action>
        </policy>
        <policy>
            <id>3</id>
            <name>HTTP-flood</name>
            <rule>cip[0].pv>100 and cip[0].rtime>60</rule>
            <action>ban</action>
        </policy>
        <policy>
            <id>1</id>
            <name>traffic attack</name>
            <rule>cip[0].pv>5 and cip[0].rlength/cip[0].pv>10240</rule> <!-- [slSlmlMlhlH] -->
            <action>ban</action>
        </policy>
        <policy>
            <id>3</id>
            <name>http slow attack</name>
            <rule>cip[0].pv>5 and cip[0].rtime/cip[0].pv>2</rule>
```

```
            <action>ban</action>
        </policy>
        <policy>
            <id>6</id>
            <name>code attack</name>
            <rule>cip[0].pv>100 and cip[0].4xx>20 and cip[0].4xx/cip[0].pv>0.6</rule>
            <action>ban</action>
        </policy>
    </domain>
    <actions>
        <action>
            <name>ban</name>
            <url>http://127.0.0.1:9000/ngproxy/rule/add</url>
        </action>
        <action>
            <name>none</name>
        </action>
        <action>
            <name>warn</name>
            <url>http://127.0.0.1:9000/ngproxy/rule/</url>
        </action>
    </actions>
</hfw>
```